IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of: John R. Chase	Attorney Docket No.: ALTRP098/A1185
Application No.: 10/693,546	Examiner: Lo, S.
Filed: October 23, 2003	Group: 2128
Title: METHODS AND APPARATUS FOR AUTOMATED TESTBENCH GENERATION	Confirmation No: 3624

CERTIFICATE OF EFS-WEB TRANSMISSION
I hereby certify that this correspondence is being transmitted electronically through EFS-WEB to the Commissioner for Patents, P.O. Box 1450 Alexandria, VA 22313-1450 on January 31, 2008.

Signed: /Joyce L. Ferreira/
Joyce L. Ferreira

PREAPPEAL BRIEF REQUEST FOR REVIEW

Mail Stop Amendment
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Applicant requests review of the final rejection in the above-identified application. No amendments are being filed with this request.

This request is being filed with a Notice of Appeal.

The review is requested for the reasons stated below.

REMARKS

Claims 1-27 are pending. Claims 1-13 and 17-27 including independent claims 1, 17, and 25 were rejected under 35 U.S.C. 103(a) as being unpatentable over Zaidi (2002/0038401 A1) in view of Heinkle (2004/0015739 A1) in view of Whitten (5805795). Claims 25-27 are rejected as being directed to software per se.

The Examiner rejected independent claims 1, 17, and 25 under 35 U.S.C. 103(a) as being unpatentable over Zaidi in view of Heinkel in view of Whitten. However, Zaidi and Heinkel and Whitten, even if appropriately combined, do not teach or suggest generating a plurality of test designs that allow testing of a design automation tool. Zaidi combined with Heinkel is believed to describe only testing of test designs, not a design automation tool. For example, Zaidi and Heinkel only describe testing of a particular physical block or a single device under test (DUT). By contrast, the claims recite selecting various physical blocks probabilitistically for a design automation tool to generate a design. According to various embodiments, the physical blocks are assumed to be correct. Any incorrect output from the design automation tool is determined to be an error in the design automation tool. A design automation tool may be able to generate a correct output for a processor A and a memory B, or a processor C and a memory D, but may not be able to handle a processor C and a memory E. Consequently, the claims recite providing numerous components to a design automation tool to allow generation of designs. By contrast, the references cited by the Examiner only contemplate providing numerous tests or test scripts.

The Examiner argues that Zaidi describes generating multiple test designs in paragraph [0037]. The Applicants respectfully disagree. Zaidi describes in paragraph [0037]. "Most of the block design directories containing RTL source code have a separate subdirectory structure underneath, e.g., "cpubr/", "cpumem/", "dma/", "intctl/", "lcd/", "mc/", "palmbus/", "pio/", "sysctl/", "timer/", and "uart/". The "<block>/sim/" subdirectory includes the simulation tests for the design. The "<block>/synop/" subdirectory includes the Synopsys synthesis scripts and output files for the design. The "<block>/vlog/" subdirectory includes the Verilog RTL source code for the design; if the embodiment language were VHDL, the "<block>/vhdl/" subdirectory includes the VHDL RTL source code for the design."

Paragraph [0037] only describes where the source code for particular components is located. The Zaidi paragraph [0037] does not teach or suggest anything about generating multiple designs or generating test designs.

The Examiner further points to paragraphs [0040]-[0041]. "Each design block has its own separate simulation directory, defined as "<block>/sim/" in the directory structure. Such directory includes all of the tests exercising that given block in the system. Simulations for the

block can be run directly from this directory. Additional tests for the block can be placed into this directory and simulated. New blocks can be easily added into the same environment by adding the same directory structure consisting of the "<newblock>/vlog/" or "<newblock>/vhdl/", and "<newblock>/sim/" directories. Tests exercising this new block in the system would be placed in the "<newblock>/sim/" directory and executed from that directory."

The \sim\ directory includes tests exercising this new block in the system. However, there is still no teaching in Zaidi of generating multiple test designs that allow testing of a design automation tool. Not only does Zaidi not teach or suggest generating the plurality of files in the /sim/ directory, but the files in the /sim/ directory are not a plurality of test designs. The Zaidi /sim/ directory includes multiple files used for testing a block. The Examiner's implication is that the multiple files for testing the block are generated. However, even if these multiple files are generated, these files are not "a plurality of test designs for testing the design automation tool." Zaidi is believed to provide multiple scripts, not test designs. The multiple scripts test only physical blocks such as "any blocks that interact with the main system buses or each other. [0040] Zaidi only possible describes simulations or scripts in the /sim/ directory. There are no generated multiple test designs.

The Examiner also argues that Heinkel teaches generating multiple test designs in paragraphs [0057] - [0060]. However, paragraphs [0057] – [0060] only describes a single "device under test 50." Heinkel in fact does not teach generating a plurality of test designs because Heinkel is configured to test a single DUT (device under test) such as a single ASIC.

By contrast, various embodiments of the present invention allow generation of multiple test designs to allow testing of a design automation tool. In one example, a test design can include a processor, a DSP core, a timer, and a network interface while another generated test design includes a processor, a cryptographic core, and a PIO, and a network interface. Top level modules are instantiated, submodules are parameterized, and interconnection logic is provided for each generated test design. It is respectfully submitted that neither Heinkel nor Zaidi teach or suggest these elements recited in the independent claims.

According to various embodiments, there are test scripts and simulations that are used to test physical blocks. Both Heinkel and Zaidi possibly describe scripts and simulations that are used to test a physical block or a "new block." By contrast, the independent claims recite generating multiple test designs that allow testing of a design automation tool.

The Examiner acknowledges that Zaidi and Heinkel even if appropriately combined do not teach of suggest using any probability function and particularly do not teach or suggest using any probability function to select modules of different types from the library as recited in

independent claims, where the modules may be processors and memory interconnected using I/O. The Examiner relies on Whitten to teach or suggest this recitation. The Applicants respectfully submit that Whitten resides in an unrelated field. The Examiner argues that Zaidi, Heinkel, and Whitten are analogous art because they are from the same field of endeavour, test design generation. The Applicants respectfully disagree. Whitten explicitly states that the invention is a computer program product that allows generation of "software test routines" to test another computer program product. [Summary] Zaidi and Heinkel by contrast, relate to running test scripts to test a physical block or a device under test. The software engineers with expertise relating to software test routines for testing computer program products generally do not have the expertise to test physical blocks or devices. These tasks are generally performed by different engineers, often in different divisions, and frequently in entirely different companies. In addition, none of the references relate to testing a design automation tool for generating designs for implementation on a physical device as recited in the independent claims.

Furthermore, Zaidi and Heinkel teach away from using a probability function. Zaidi and Heinkel provide a deterministic test of a particular test design. The purpose of Zaidi and Heinkel is to test particular physical blocks. Hypothetically, if these physical blocks were probabilistically selected, this would not provide any additional efficiency during testing of the physical blocks. Some physical blocks would be redundantly tested while other physical blocks would remain untested. By contrast, the techniques of the present invention contemplate testing a design automation tool. In order to sufficiently test a design automation tool, numerous blocks are probabilistically selected to create numerous designs. According to various embodiments, the numerous blocks in the library are already thoroughly tested. Any error in the design automation tool output for designs using probabilistically selected components would be the result of a design automation tool error. For example, the design automation tool may be able to handle Processor A and Memory B in a system but may not provide the appropriate output for Processor C and Memory D.

The Examiner rejected claims 25-27 as directed to software per se. The Applicants respectfully disagree that claims 25-27 are directed to software per se. Claims 25 recite "processing means" and "interface means." These means elements provide structural and functional interrelationships between functional elements and the rest of a system such as a computing processing system, provide a "useful, tangible, and concrete result" and are believed to be statutory. Furthermore, the invocation of USC 112(6) is not believed invalidated simply because the Specification states that "For example, various aspects described above may be

implemented using firmware, software, or hardware. Aspects of the present invention may be employed with a variety of different file formats, languages, and communication protocols and should not be restricted to the ones mentioned above." The broadening language is not intended to claim everything, but instead merely states not every possible embodiment can be described.

In light of the above remarks, the rejections to the independent claims are believed overcome for at least the reasons noted above. Applicants believe that all pending claims are allowable in their present form. Please feel free to contact the undersigned at the number provided below if there are any questions, concerns, or remaining issues.

Respectfully submitted,
BEYER WEAVER LLP

/G. Audrey Kwan/
G. Audrey Kwan
Reg. No. 46,850

P.O. Box 70250
Oakland, CA 94612-0250
(510) 663-1100

APPENDIX: PENDING CLAIMS

1. (Previously Presented) A method, comprising:

generating a plurality of test designs, the plurality of test designs having varied characteristics to allow testing of a design automation tool, wherein generating one of the plurality of test designs comprises:

instantiating the I/O structure of a top level module, the top level module having input and output pins;

selecting a plurality of submodules from a design module library, wherein a probabilistic function is applied to select submodules of different types from the library;

parameterizing the plurality of submodules from the design module library for interconnection with the top level module, the plurality of submodules having input and output lines;

providing logic to interconnect the plurality of parameterized submodules as well as to connect the plurality of parameterized submodules to various input and output pins of the top level module;

applying the plurality of test designs to test the design automation tool.

2. (Previously presented) The method of claim 1, wherein the design automation tool is used to implement hardware descriptor language designs on a programmable chip.

3. (Previously presented) The method of claim 1, wherein the design automation tool is used to implement designs on an ASIC.

4. (Previously presented) The method of claim 1, wherein instantiation constraints are used to select the plurality of submodules.

5. (Previously presented) The method of claim 1, wherein the design automation tool is a synthesis or a place and route tool.

6. (Previously presented) The method of claim 1, wherein providing logic to interconnect the plurality of parameterized modules comprises identifying inputs and outputs.

7. (Previously presented) The method of claim 6, wherein inputs comprise input pins of the top level module, submodule output lines, and registers.

8. (Previously presented) The method of claim 6, wherein outputs comprise output pins of the top level module, submodule input lines, and registers.

9. (Previously presented) The method of claim 8, wherein providing logic to interconnect the plurality of parameterized modules further comprises classifying inputs and outputs as clock lines, control lines, and data lines.

10. (Previously presented) The method of claim 8, wherein generating one of the plurality of test designs further comprises:

generating randomized logic.

11. (Previously presented The method of claim 10, wherein randomized logic is generated to drive outputs.

12. (Previously presented) The method of claim 10, wherein generating randomized logic comprises directly wiring outputs to inputs, generating a logic expression using inputs, generating a mathematical expression using inputs, or generating decision logic.

13. (Previously presented) The method of claim 6, wherein parameterizing the plurality of submodules comprises defining interfaces, data width, and the type of signal for input and output lines associated with the submodule.

14. (Previously presented) The method of claim 6, wherein submodules comprise adders, phase lock loops, memory, and timers.

15. (Previously presented) The method of claim 6, wherein generating one of the plurality of test design further comprises selecting a clock structure for each output.

16. (Previously presented) The method of claim 15, wherein clock structures include a plurality of synchronous and asynchronous structures.

17. (Previously Presented) A computer system, comprising:

memory operable to hold information associated with a design module library;

a processor coupled to memory, the processor configured to generate a plurality of test designs, the plurality of test designs having varied characteristics to allow testing of a design automation tool, wherein generating one of the plurality of test designs comprises:

instantiating the I/O structure of a top level module, the top level module having input and output pins;

selecting a plurality of submodules from the ~~a~~ design module library, wherein a submodules of different types are randomly selected from the library;

parameterizing the plurality of submodules from the design module library for interconnection with the top level module, the plurality of submodules having input and output lines;

providing logic to interconnect the plurality of parameterized submodules as well as to connect the plurality of parameterized submodules to various input and output pins of the top level module;

applying the plurality of test designs to test the design automation tool.

18. (Original) The computer system of claim 17,wherein the design automation tool is used to implement hardware descriptor language designs on a programmable chip.

19. (Original) The computer system of claim 17,wherein the design automation tool is used to implement designs on an ASIC.

20. (Original) The computer system of claim 17,wherein the design automation tool is an electronic design automation tool.

21. (Original) The computer system of claim 17,wherein the design automation tool is a synthesis or a place and route tool.

22. (Original) The computer system of claim 17, wherein providing logic to interconnect the plurality of parameterized modules comprises identifying inputs and outputs.

23. (Original) The computer system of claim 22, wherein inputs comprise input pins of the top level module, submodule output lines, and registers.

24. (Original) The computer system of claim 22, wherein outputs comprise output pins of the top level module, submodule input lines, and registers.

25. (Previously Presented) An apparatus for generating test a testbench, the apparatus comprising:

processing means for generating a plurality of test designs, the plurality of test designs having varied characteristics to allow testing of a design automation tool, wherein means for generating one of the plurality of test designs comprises:

means for instantiating the I/O structure of a top level module, the top level module having input and output pins;

means for selecting a plurality of submodules from a design module library, wherein a probabilistic function is applied to select submodules of different types from the library;

means for parameterizing the plurality of submodules from the design module library for interconnection with the top level module, the plurality of submodules having input and output lines;

means for providing logic to interconnect the plurality of parameterized submodules as well as to connect the plurality of parameterized submodules to various input and output pins of the top level module;

interface means for applying the plurality of test designs to test the design automation tool.

26. (Previously presented) The apparatus of claim 25, wherein the design automation tool is used to implement hardware descriptor language designs on a programmable chip.

27. (Previously presented) The apparatus claim 25, wherein the design automation tool is used to implement designs on an ASIC.